UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/717,187	11/20/2000	Vikram Joshi	50277-0378	9515

29989 7590 04/08/2004

HICKMAN PALERMO TRUONG & BECKER, LLP
1600 WILLOW STREET
SAN JOSE, CA 95125

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2124	12

DATE MAILED: 04/08/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/717,187	JOSHI ET AL.	
	Examiner	Art Unit	
	Tuan A Vu	2124	

-- The MAILING DATE of this communication app ars on the cover sheet with th correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09 February 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *20 November 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 2/9/2004.

As indicated in Applicant's response, claims 18-24 have been amended. Claims 1-24 are pending in the office action.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gorshkov et al., USPN: 6,490,721 (hereinafter Gorshkov), in view of Bowman-Amuah, USPN: 6,442,748 (hereinafter Bowman) , and further in view of Khoyi et al., USPN: 5,303,379 (hereinafter Khoyi).

As per claim 1, Gorshkov discloses debugging method comprising:

preserving a memory state of a preserved portion of a first software program (e.g. *fork* -- col. 3, lines 54-60 – Note: forking a child is equivalent to preserving a parent program, or first program);

dynamically linking a second software program to the first program without de-allocating the first program from volatile memory (e.g. step *10* – Fig. 2; col. 4, lines 29-33 — Note: the linking of the child to the target user program being a copy of the parent is equivalent to linking second program to first);

executing the second program (e.g. step *12* – Fig. 2; Fig. 4), the second program when executing would cause modifications to the preserved portion of the first software program (e.g. steps *21, 30* -- Fig. 4 – Note: child executing a patch call with linking re-entry calls affecting the preserved portion in the parent code is equivalent to second program would otherwise cause modifications to the preserved portion of the parent); and

making copy of such preserved portion of the first software (e.g. step *6* – Fig. 2 – Note: the to-be-modified areas of parent code being duplicated in child amount to making copy of targeted area) and effect the patch modification to said copy before linking the patch code back into the targeted data (e.g Fig. 2-4).

But Gorshkov does not specify if execution of the second software would otherwise cause modification to targeted data that is preserved portion of the first software program, then making a copy of the targeted data and modifying the copy of the targeted data to generate a modified copy of the targeted data without modifying the targeted data in the preserved portion of the first software. By disclosing making copy of the parent process, i.e. a child process, so to incorporate therein the subprogram calls using a patch area (steps *22-28* -Fig. 4) without affecting the parent process; i.e. executing the user defined subprograms replacement at specific path points branched off (step *21*- Fig. 4)from the child process before branching the execution control back into the child program (steps *29-30* – Fig. 4), Gorshkov has disclosed making copy of the preserved portion of the first software program and modifying the copy of the first software without modifying the targeted data of the preserved first software. Moreover, Gorshkov informs about a time-efficient debug technique to overcome one potential bug issue inherent to simultaneous changes effected by team developer (col. 1, lines 32-47). In the

scenario where more than one users are involved with code modifications to targeted data of a parent code, the use of copy of a main target code so to incorporate changes or modifications to such copy before reconciling the changes made at the copy into the preserved target code was a well-known concept in software code modifications and simultaneous team development actions as suggested by Gorshkov at the time the invention was made. In a method for extensive code modification and integrating environment with code versioning for accommodating multiple developers or publishers reminiscent to the code modification and concurrent code change feared by Gorshkov, Bowman discloses control of ownership over the target code to change and making copy of the target code in each instance of change per publisher or developer without stopping the ongoing main process (e.g. col. 177, lines 38-58; col. 293, lines 32-56; Fig. 177-178). Further, Khoyi, via a method for integrating changes of data between child processes and parent processes analogous to the debugging child/parent paradigm by Gorshkov, discloses using of specialized structures to link objects targeted in the modifications schemes and thus making copies of targeted portion or object from thus linked objects so as to effect modification therein (e.g. col. 3, lines 33-44; col. 38, lines 7-22). In the context that more than one developers are to modify a target code as suggested by Gorshkov, it would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the code patching such as taught by Gorshkov so to implement whenever resources are available the technique of creating additional copies of the targeted portion and effect all the changes to the determined targeted data therein prior to integrating such changes back into the main target code as suggested by Bowman and enhanced by Khoyi, because this would eliminate the shortcomings of having unresolved code references due to concurrent code changes as feared by Bowman and maintain a integrity of the

original code for securing roll-outs (e.g. Bowman: col. 177, line 60 to col. 178, line 43) as well as provide session independency while modifying the copy at will (e.g. Khoyi: col. 2, lines 4-17; *remain essentially independent* - col. 3, lines 33 to col. 4, line 42).

As per claim 2, Gorshkov does not disclose publishing the preserved portion of the first program with a symbolic name associated it with the second program; nor does Gorshkov disclose accessing the second program by multiple users via the symbolic name. Gorshkov, however discloses the contention due to multiple users trying to modify code for bugs and the desirability to accomplish such code change in a uninterruptive and more controlled manner (e.g. col. 1, lines 27-47). A desired resolution to synchronizing multiple and simultaneous changes applied to a persistent storage or shared source of data as suggested by Gorshkov is evidenced by Bowman. Bowman suggests the use of versioning services for check-in/check-out of instance or copies of objects (e.g. Fig. 14; col. 53, lines 58 to col. 54, line 48; *reference this instance* -- col. 286, lines 21-45), hence suggested use of identifier (version number) to reference a preserved code targeted for changes. It would have been obvious for one of ordinary skill in the art at the time the invention was made to provide the users with publishing means which would enable user access to the second program the corresponding preserved portions in the first program, i.e. using the technique suggested by Bowman to link users to the latest representation of the recently updated source data, i.e. version identifier, because this would enable up-to-date data access by users as well as imparting user's independency of tasks without risk of infringing upon someone's else state of task achieved, thereby avoiding memory reference faults through misdirected retrieval/read of obsolete or non-existent or shared data.

As per claim 3, in the combination of Gorshkov/Bowman of claim 1, Bowman does teach a database services (e.g. Fig. 12, 14, 156-161) and this database limitation would have been obvious in view of the combined teachings by Gorshkov and Bowman as mentioned in claim 2. One of ordinary skill in the art at the time the invention was made would be motivated to modify the source code (first program) to upgrade by Gorshkov to make it a database-related program because both Gorshkov and Bowman desire to resolve multiple upgrade to the data source and expediently propagate the changes to provide persistency to the common source while providing fault-free up-to-date data use, using the upgrade technique applied to a copy such as suggested by Bowman; and applying the technique of dynamic creating of replica for expediting modification before committing as such is one typical operation in a database upgrade application.

As per claim 4, Gorshkov does not teach a database system; but this limitation has been addressed in claim 3 above and would have been obvious herein for the same rationale. On the ground that Bowman teaches about a database being updated, Gorshkov further discloses preserving of the memory state of the preserved portion of the first program (e.g. steps 22-24 – Fig. 4; steps 35-37 – Fig. 5 – Note: the context switching applied to child code being a copy of the first program is equivalent to saving state of the first program, and the need to upgrade or patch). This, in conjunction with Bowman, is implicitly teaching that a portion in a database application has failed or requires immediate interruption for consistency resolution.

As per claim 5, Gorshkov does not specified accessing the modified copy of the targeted data upon a subsequent attempt. Official notice is taken that version control software limiting access by a registered user to just the modified copy of a preserved and version-controlled

document was a well-known concept in the art at the time the invention was made (e.g. PVCS, Sybase). In view of this well-known concept, this instant limitation would have been obvious by virtue of the combined teachings by Gorshkov, Bowman (e.g. col. 176, line 48 to col. 177, line 58) and Khoyi using the rationale from claims 1-2 above because the preserved portion of the targeted data has now been redirected via a symbolic reference (e.g. versioned copy) to the updated portions applied to the copy of such targeted data.

As per claim 6, this limitation would have been obvious by virtue of the combined teachings by Gorshkov, Bowman and Khoyi above because the process of dynamically linking by Gorshkov is thereby further enhanced so as to becoming a dynamic database persisting via replication and the teachings by Bowman would have combined to render the subsequent access by an user limitation obvious as per the same rationale used in claim 2 above.

As per claim 7, Gorshkov discloses dynamic linking and executing of the second software program by a user (e.g. col. 4, lines 29-33; step *12* – Fig. 2; Fig. 4); a first modified copy of the targeted data (e.g. step *6* – Fig. 2). But Gorshkov does not specify that if the steps of executing an operation that would cause modification of targeted data, then performing such operation by making a second copy and modifying a second copy being separate from a first modified copy; nor does Gorshkov explicitly disclose that the first modified copy has been linked and executed by a first user. As addressed in claim 1, the process allowing the child process to link to the parent process has been met by Gorshkov as far as making modifications to the copy without affecting the first software and the targeted data to be modified in the first software has been interpreted as data being modified in the eventuality of multiple copies of parent code being modified by simultaneous users prior to reconciliation. By virtue of the

combined teachings by Gorshkov, Khoyi, and particularly Bowman, these steps limitations to link to execute the second software, to generate a modified copy by a first user; then repeat the process by creating a modified second copy by another user would have been obvious because of the desired intent by Gorshkov/Bowman/Khoyi as not to allow users to access undesired or obsolete/unallocated areas of the data stored, thus creating for each user an instance (e.g. Bowman: *check-in/out* - re claim 2) which would direct him/her to an instance of claimed/appropriate areas of the latest modified target code; and that the propagation of the data changes should be available to all users as intended by Bowman (re claim 2), with the repeated scenario of informing each user with a reference to the latest portion of data to start the modification with.

As per claim 8, the steps as claimed performed by a third user would have been obvious in view of the rationale used to reject claim 7; hence are rejected herein using the same ground of rejection used in claim 7 above.

As per claim 9, this is the computer medium version of claim 1 above, hence is rejected herein using claim 1 rejection; further using the disclosure by Gorshkov (col. 5, lines 14-23) to address the computer-readable medium.

As per claim 10, this is the computer medium version of claim 2 above, hence is rejected herein using claim 2 rejection.

As per claims 11-12, these claims correspond to claims 3-4, respectively, hence incorporate each the corresponding rejection as set forth therein.

As per claims 13-16, these claims correspond to claims 5-8, respectively, hence incorporate each the corresponding rejection as set forth therein.

As per claim 17, Gorshkov discloses an apparatus for debugging a first software program, such apparatus comprising a memory storing instructions operable for performing the steps of:

preserving (first software program without de-allocating the first);

dynamically linking (second software program);

executing the second program; and

making copy (targeted data) and effecting modifications to the first software

just as recited in claim 1 above.

These steps are rejected herein with the corresponding rejections as set forth therein.

But Gorshkov does not specify making a copy of the targeted data and modifying the copy of the targeted data to generate a modified copy of the targeted data without modifying the targeted data in the preserved portion of the first software. This limitation has been addressed in claim 1 above and is rejected herein with the same rationale as set forth therein using Bowman and Khoyi's combined teachings.

As per claims 18-24, these claims are the apparatus versions of claims 2-8, respectively; hence are rejected using the corresponding rejections as set forth respectively therein.

Response to Arguments

4. Applicant's arguments filed 09/29/2003 have been fully considered but they are not persuasive in view of the current grounds of rejection. Following are the reasons therefor.

(A) Applicants have submitted that Gorshkov does not teach or suggest modifying the child process; and that Gorshkov teaches modifying the parent when the child process patches the debugging routines into the parent process (Appl. Remrks, pg. 13, 2nd para). It is noted that

Gorshkov teaches replacing the user defined subprograms in the patch area from within a child process and incorporating all the replaced instructions by the child process and linking such patch subprogram calls into the parent process; thus has disclosed making copy of the preserved portion of the parent process (or child process) and making modifications to the address information of the copy of such preserved portion, i.e. modification to the child process registers, without affecting the parent process. The claim recites 'if execution of the second program would otherwise cause modification to targeted data that is in preserved portion of the first software program, then making a copy of the targeted data and modifying the copy of the targeted data to generate a modified copy of the targeted data without modifying the targeted data in the preserved portion of the first software'. Such limitation is interpreted according to the understanding that only if the child process would cause modification to a some data known to exist in the parent process (such 'targeted data' which is not explicitly evidenced from Gorshkov's linking of child process back into parent process), then a copy of such data would be made and modification to such copy would be implemented. As recited, the modification of the so-called targeted data in the preserved portion of the first software was not defined sufficiently to point out which data of the preserved first software portion would qualify as targeted data. Hence, it is unclear as to which data in the parent process as shown by Gorshkov would be called targeted data and thus susceptible for modification; and according to a broadest and reasonable interpretation, such targeted data would be any data in the first software that require modification or reconciliation in the light of existing changes effected by the second software. The rejection thereby addresses only the limitation that a copy of the preserved portion of the first software is made and modification to the execution state or control state of such copy (see Fig. 4) is effected

without affecting the preserved portion of the parent or first software; and Gorshkov has met such limitation. The recited 'modification of targeted data that is in the preserved portion of the first software' is therefore not to be (and has not been) equated to 'modifying the parent when the child process patches the debugging routines into the parent process' as alleged by Applicants (Appl. Remrks, pg. 13, 2nd para); or the linking to the first software --or parent process-- by Gorshkov's invoking calls to the user defined subprograms, because as interpreted no explicit targeted data is modified in Gorshkov's linking to the parent process for subprograms to be invoked. As to the limitation that a modification that would otherwise modify targeted data of a preserved portion of the first software program, the rejection has pointed to Gorshkov's desire to prevent conflicts due to simultaneous intervening modification of same code under debugging, and how the teachings by Bowman, Khoyi, and well-known concepts in the art combine with Gorshkov's suggestion to meet the limitation. In the scenario that targeted data in a first software is to be modified by more than one developers, then the implementing of more than one copies to incorporate the changes and then to reconcile all such copies after changes have been made back into the first software would have been obvious as set forth in the rejection. Hence, Gorshkov has met the limitation that a copy of the first software is made so to incorporate the modifications therein without affecting the preserved portion of the first software. The linking of the user defined code into the first software is not at odds with the limitation that Applicants refer to as 'if execution of the second program would otherwise cause modification to targeted data ..., then making a copy of the targeted data and modifying ... preserved portion of the first software' because no 'targeted data' in the first software is explicitly affected by such linking. As mentioned above, the 'targeted data of the preserved portion of the first software' has

been interpreted only to fit the context of multiple users' simultaneous targeting one parent code to modify code with data targeted as to-be-modify data. Thus, the combination as set forth in the claim has addressed the scenario in which copies of the parent process can be made to incorporate all changes to targeted data by different users/developers and that a reconciliation of all such changes (of targeted data) into the parent code be effected to avoid conflict issues; and such combination would have been obvious as set forth in the rejection in such 'if' conjecture.

(B) Applicants have submitted that 'the target program contained in the parent process would not contain calls to the debugging routines... unsatisfactory for its intended purpose' (Appl. Remrks, pg. 14, 2nd para). As pointed out earlier, the modifying of Gorshkov's approach to implement multiple copies for applying dynamic and simultaneous modifications and reconciling such separate copy changes into the parent process is only used when addressing cases where Gorshkov's debugging method is being implemented in a multiple and concurrent users environment, environment such as suggested by Bowman and Khoyi in which some targeted data in the parent code are specified to be modified, and in which resources are such to enable the creation of as many copies as the concurrent users situation demand. The current method used by Gorshkov should be sufficient to meet the required creation of a copy of the parent process; and effecting of modifications therein without affecting some 'targeted data' because as cited, the dynamic linking of subprograms by the child process into the parent process clearly does not modify any 'targeted data' in the 'preserved portion' of the first software. As mentioned above, the linking of the child settings into the parent process does not teach away what Applicants regard as modifying targeted data in the preserved portion of the first software. Hence, there is no taking away the debugging routines by Gorshkov's. The invoking of debugging code

dynamically, i.e. without interrupting the first software, by means of creating a copy thereof and making changes in the copy process, such changes affecting specific data in the copy process, would still have served the intended purpose of Gorshkov's method. And the eventuality in which targeted data are identified for modifications as suggested by Bowman and Khoyi, and where more copies are to be generated to provide modifications and subsequent incorporation of all changes back into the parent code after all targeted data would have been affected from within the children processes would still have preserved the purpose of Gorshkov's method if resources were available as suggested by Bowman and Khoyi. The reconciling of all separate copies changes into the main parent program, the debugging calls by Gorshkov without affecting any 'targeted data' would have been uninterrupted, thus not rendering Gorshkov's method inapposite in regard to its intention.

In view of the above, the rejections will stand as set forth above.

Conclusion

8. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event,

however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 872-9306 (for formal communications intended for entry)

or: (703) 746-8734 (for informal or draft communications, please label "PROPOSED" or "DRAFT" – please consult Examiner before use)

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA. , 22202. 4th Floor(Receptionist).

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

VAT
March 28, 2004

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100